Другие языки:
English • русский

Содержание

Разработка модуля

1. Определения

Модуль - это компонент платформы, который позволяет управлять одним типом оборудования по его уникальному идентификатору. Модуль расположен в магазине модулей.

Управляющая панель (мобильное устройство) - устройство, с которого запускается проект i3 lite.

Статический модуль - это модуль, обладающий фиксированным (не расширяемым при запуске или в процессе работы) функционалом.

Динамический модуль - это модуль, обладающий динамическим (расширяемым при запуске или в процессе работы) функционалом, такие модули считаются сложными в реализации, но они более гибкие в использовании

Магазин модулей - это облачный сервер для загрузки готовых модулей.

iRidium Studio - редактор, который позволяет создавать модуль управления устройством для приложения i3 lite.

i3 lite - это приложение для создания, настройки и использования проектов i3 lite.

Проект i3 lite – проект автоматизации, составленный из готовых модулей управления оборудования. Отличие i3 lite проекта от обычного проекта i2 Control в том, что проект i3 lite может изменяться динамически без предварительного изменения в iRidium Studio. i3 lite проект состоит из небольших частей и при помощи встроенного конструктора в приложении i3 lite может собираться в единое целое и легко редактироваться. Пропадает ряд промежуточных процессов, таких как установка программ на ПК, загрузка проекта на панель. Все происходит в одном месте - на панели.

SubDevice (под-устройство) - это виртуальное представление рабочей зоны физического устройства.

Комната (в i3 lite) - визуальная часть i3 lite, содержащая виджеты управления оборудованием.

Widget - визуальный компонент модуля, размещенный в комнате, имеет ограничения по размерам и фиксированную позицию в интерфейсе комнаты. Содержит основной функционал модуля. Для дополнительного функционала используются Remote.

Remote - визуальный компонент модуля, который по умолчанию скрыт и открывается только при нажатии на соответствующий элемент widget'a. Не имеет ограничений по размеру (не более размера дисплея панели) и может располагаться в любой части интерфейса комнаты (обычно располагается посередине дисплея панели). Содержит дополнительный функционал модуля.

Макрос - последовательность управляющих команд различных модулей, которая активируется нажатием. Содержит Action (Действия) и Condition (Условие).

Сценарий - последовательность управляющих команд различных модулей, которая активируется автоматически при срабатывании определенного события. Содержит Event (Событие), Action (Действие) и Condition (условие).

Event (Событие) - компонент сценариев, описывает управляющую команду и границы значения. При отправке значения входящего в указанные границы выполнится сценарий

Action (Действие) - компонент макросов и сценариев, описывает управляющую команду и значение, которое должно на неё (команду) отправиться. Бывает 2 типов: Simple Action и Advanced Action.

Simple Action - простой Action используется когда заранее известна команда и значение Action'a.

Advanced Action - расширенный Action используется, когда заранее известна команда, но неизвестно значение. Значение задается динамически при использовании Action'a в i3 lite (но не при создании в iRidium Studio).

Condition (Условие) - компонент макросов и сценариев, описывает управляющую команду и границы значения. При отправке значения входящего в указанные границы может выполняться Action.

Scanner (сканер) - это программная часть, которая анализирует какую-либо шину на подключенные устройства. Подключенные устройства формируются в список, для выбранного устройства качается модуль и становится доступен в i3 lite.

2. Как начать

Для начала рекомендуется ознакомиться с документацией iRidium. В ней описаны основы работы в iRidium. В i3 lite документации описываются особенности при разработке модулей под i3 lite платформу, рассчитанные на пользователя знакомого с iRidium. При написании скрипта используйте i3 lite API.

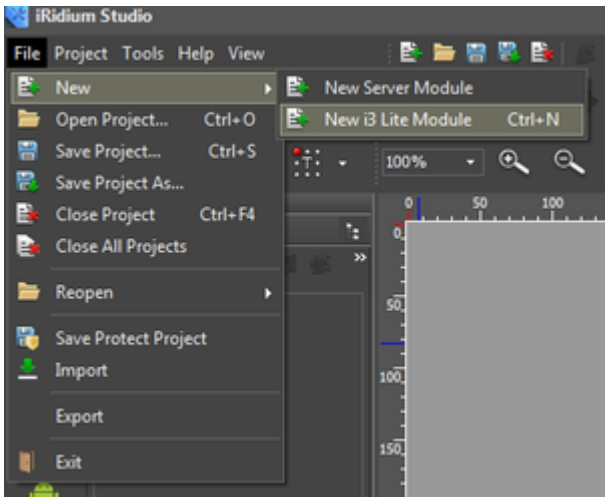
Рассмотрим из каких частей состоит жизненный цикл модуля:

- Разработка графической части в iRidium Studio
- Разработка драйверной части (скрипты, устройства)
- Тестирование модуля
- Публикация в магазине (iRidium Store)
- Использование (Загрузка модуля через приложение i3 из облака и добавление в проект автоматизации)

Создание модуля происходит в редакторе iRidium Studio.

Модуль включает в себя: графическую, драйверную и скриптовую части. Модуль может быть использован на сервере, без каких либо доработок.

Создание модуля.
Чтобы создать модуль, запустите редактор, выберите пункт File -> New -> New i3 lite Module.



При создании модуля указывается: имя проекта и стиль, в котором будет создан модуль, в дальнейшем стиль может быть изменен пользователем. После этого будет создан проект модуля, и можно приступать к разработке. Следует учесть, что модуль может работать как на смартфоне, так и на планшетном компьютере.

3. Структура модуля

Редактор iRidium Studio предоставляет следующий функционал для создания панельного

модуля: создание SubDevice; создание Widget; создание Remote; создание Actions, events, conditions и скриптовую часть .

3.1. SubDevice

Из определения понятно, что SubDevice это виртуальное представление рабочей зоны физического устройства. Многие устройства поддерживают несколько рабочих групп с одинаковыми функциями. Каждая рабочая группа может находиться в отдельной зоне. Для полноценного управления зоной необходимо создавать SubDevice для каждой рабочей группы устройства. Например 6-канальный диммер HDL, содержит в себе 6 диммируемых каналов, с помощью которых можно управлять 6 рабочими группами. Поэтому целесообразно выделить в модуле 6 SubDevice, для возможности создания действий, событий и условий для каждой рабочей группы по отдельности
Наличие SubDevice в i3 lite модуле обязательно и их может быть от 1 до множества. Обычно количество SubDevice равно количеству каналов контроллера.

Рассмотрим из чего состоит **SubDevice**:

1. Условий (Conditions)
2. Действий (Actions)
3. Событий (Events)
4. Виджетов (Widgets)

Условия, действия и события являются логическими элементами SubDevice, на их основе создаются макросы и сценарии (подробнее - раздел 3.4.). Widget (виджет) является графическим элементом, с помощью него пользователь будет управлять SubDevice. Обычно 1 SubDevice содержит 1 виджет, но количество условий, действий и событий сильно варьируется (от 0 до множества).

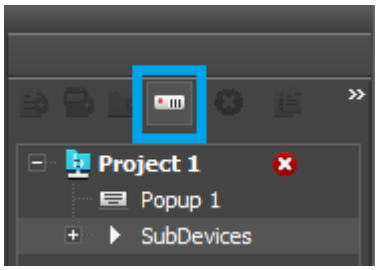
Что выбрать: скрипт или редактор?

Создать SubDevice (а также Widget, Remote, Actions, Events и Conditions) можно двумя способами: с помощью редактора и с помощью скрипта (методы смотреть в lite API). Создание с помощью редактора рекомендуется в случае, когда заранее известно количество SubDevice в модуле (статический модуль) и их немного. С помощью скрипта рекомендуется создавать SubDevice, когда их много и вручную в редакторе их добавление займет большое количество времени. Либо в случае отсутствия информации о количестве SubDevice в модуле. Количество SubDevice может варьироваться (динамический модуль).

Например, есть модуль для контроллера освещения, который можно соединить максимум с 4 рабочими группами освещения. В таком случае целесообразно добавить 4 SubDevice'а с помощью редактора. Однако, в случае модуля для DMX контроллера (контроллер для управления освещением, способными светить разным цветом), к которому можно подключить до 512 рабочих групп, создание SubDevice с помощью редактора - будет не оптимальным решением. Во-первых, добавление 512 SubDevice займёт много времени, а во-вторых излишне будет отображать 512 виджетов одновременно, многие из которых, скорее всего, использоваться не будут. В таком случае целесообразно написать скрипт, который будет создавать нужное количество SubDevice.

Создание с помощью редактора

Для создания SubDevice с помощью редактора (iRidium Studio) необходимо создать проект, далее выбрать свой проект в дереве и в нём выбрать элемент "SubDevices", после этого будет доступна кнопка добавления SubDevice.



Далее можно будет добавлять actions, events, condition и popups (widgets или remote) к соответствующему SubDevice.

SubDevice имеет следующие параметры:
Name - имя SubDevice;
Device - имя Device, которому принадлежит SubDevice (обязательный параметр для заполнения).

Как работать с SubDevice с помощью скрипта описано в разделе 5.2.

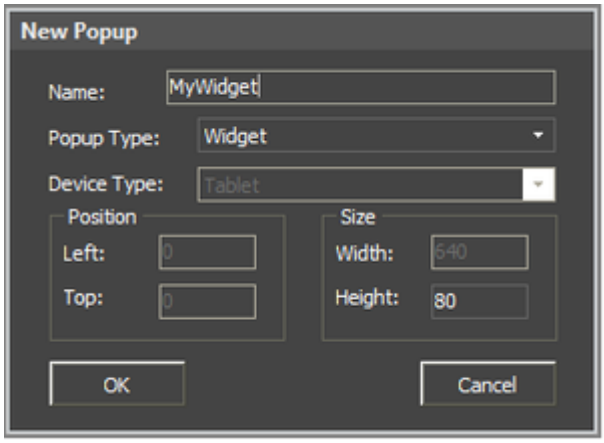
3.2. Widget

Widget (виджет), визуальный компонент модуля, разновидность Popup'а, который будет отображен в комнате. Основная задача виджета это отображение основных элементов управления SubDevice и обеспечить переход на пульт управления (Remote) SubDevice'ом модуля. Например, для управления медиа плеером можно поместить слайдер громкости или кнопку Mute, для быстрого доступа к регулировке громкости. Погодный виджет может отображать текущую погоду, а сам пульт управления будет содержать прогноз на все 5 дней. Как и без SubDevice без виджета i3 lite модуль не имеет смысла, так как не будет иметь графического представления в i3 lite проекте. Виджет ограничен размерами:

1. Ширина **640**
2. Высота от **80**

Рис. Пример виджета

Создание виджета с помощью редактора.
Выбрать созданный SubDevice в дереве и выделить ветку "Popups". Далее нажать кнопку "Add popup page" в меню над деревом проектов. В появившемся окне выбираем popup type "Widget", задаём имя и свойства:



При проектировании виджета необходимо использовать графические элементы

В i3 lite модуле можно добавлять только графические элементы из галереи.

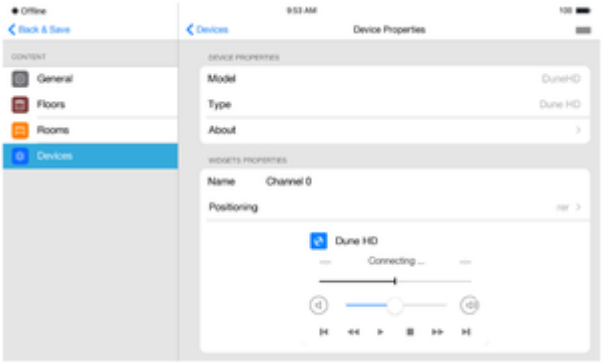
В созданном виджете можно менять параметры с помощью **Object Properties**.
Связывание графического элемента и канала происходит как в обычном Popup'е:

Рис. Связывание канала команды и графического элемента

Виджет также можно создавать и изменять с помощью скрипта (см. lite API).
Использование.
Виджет будет отображаться:

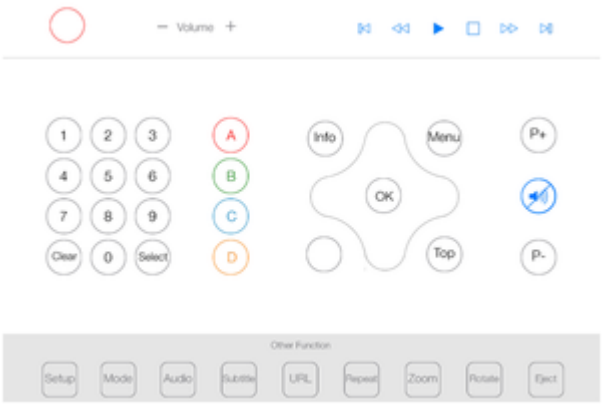
- В комнате, при использовании пользователем проекта:

- И в настройках модуля, при добавлении его в проект:



3.3. Remote

Remote (пульт управления) - это графическое пространство для управления SubDevice, в отличии от виджета Remote может иметь произвольные размеры, позицию и отображается по вызову. Пульт отображается в момент, когда на виджете была нажата кнопка перехода к пульту. На пульте управления размещаются элементы управления оборудованием (кнопки, списки, переключатели, уровни). Пультов управления может быть от 0 до нескольких. Пример пульта:



Remote создаётся аналогично виджету (раздел 3.2), с одним лишь отличием, что при нажатии "Add Popup Page" в параметре "Device Type" надо выбрать "Remote".

Так как Remote это Popup, то работа с ним происходит аналогичным образом (см.iRidium Script API). Каждый Remote должен иметь экземпляр для планшета и телефона.

3.4. Условия (Conditions), действия (Actions) и события (Events)

Условия, действия и события используются в макросах и сценариях. Механизм действий схож с графической связкой элементов и каналов драйвера, с единственной разницей, что на графический элемент обязательно надо нажать для того, чтобы драйвер выполнил команду. В случае действий, заданные значения, например 100 для канала Volume, можно будет активировать программно. Это позволяет различным драйверам взаимодействовать друг с другом. Помимо этого сохраненные действия можно комбинировать между собой и образовывать динамические сценарии во время использования приложения, которые хранятся в памяти. Сценарии можно так же поместить на графический элемент и активировать нажатием.

Количество условий, действий и событий зависит от широты функционала устройства и необходимости создания сценариев и макросов.

Создание Actions.

Существует два вида Actions: Simple - простое и Advanced - расширенное. Различаются они тем, что у Simple вся информация, необходимая для функционирования действия, известна в момент создания Action. Например, чтобы включить диммер на полную яркость - необходимо активировать канал setDimmer и отправить туда значение 255. Значение известно заранее. Advanced Action создается на основе неполной информации о предстоящем действии, в случае с диммером, можно привести пример установки диммера в любое допустимое значение от 0 до 255, но какое значение надо будет записать неизвестно, причем неизвестен даже момент, в какой требуется получить это значение.

Создание событий Events и условий Conditions
Events позволяют определить на какие значения канала SubDevice может реагировать система. Если в комнате включили свет, что соответствует установке значению 255 канала диммера, то, например, требуется закрыть штору. Events и Conditions относительно настройки ничем не отличается и их можно целиком продублировать. Отличие заключается в моменте считывания данных. Если Events отслеживаются системой постоянно и в момент принятия канала обратной связи указанного значения будет сгенерировано событие, то для Conditions значение проверяется только по требованию, например в момент срабатывания события. Conditions никогда само не активирует что-либо, но является условием выполнения Action'a. Пример использования Conditions: Если в комнате включили свет, и время больше 18:00 вечера, то закрыть шторы. Таким образом если в комнате выключить свет днем, шторы не закроются.

5. Скриптовая часть

Скриптовая часть при разработке i3 lite модуля имеет некоторые особенности. Перед разработкой скриптовой части lite модуля рекомендуем ознакомиться с iRidium Pro API и i3 lite API.

5.1. Основы

Модуль может содержать неограниченное количество скриптов и может не содержать ни одного скрипта. Существуют разные типы скриптов для модуля:

- Driver - в скриптовых файлах данного типа должны содержаться скрипты, отвечающие за работу с драйвером
- GUI - В этих скриптовых файлах содержатся скрипты, отвечающие за работу с графикой
- Common - в данных скриптовых файлах должны содержаться скрипты общего характера, тут может быть набор дополнительных функций или описание классов второстепенного назначения
- Setup - это специальны скриптово ал. в нем содержиться информация о настройке модуля. а именно - список полей, обязательных для заполнения, типы этих поле и ункции проверки правильности введенных данных.

Разделение на типы необходимо для правильной обработки модуля на сервере. На сервере работает только логика и драверна часть модуле, но сервер ничего не знает о графической части. Поэтому разработчику модуля требуется разбивать скрипты по типам. Сервер игнорирует файлы для работы с графической частью.

При разработке модуля следует использовать пространство имен module вместо IR. Таким

образом слушатель будет иметь вид module.AddListener(...

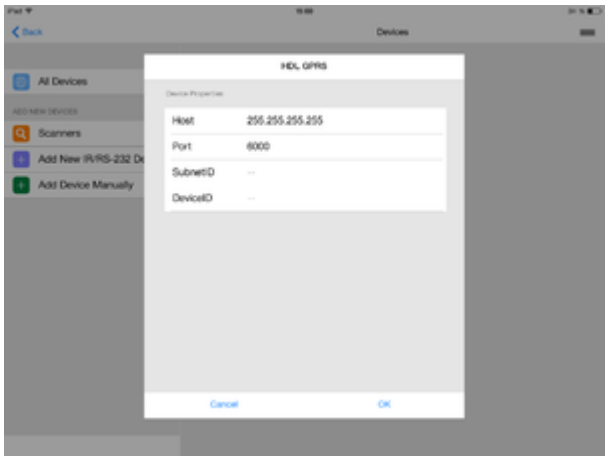
Каждый скриптовой файл должен начинаться со слушателя

```
module.AddListener(IR.EVENT_MODULE_START, 0, function(){});
```

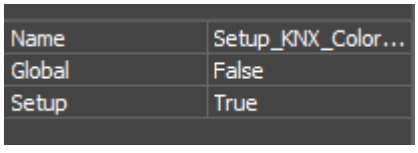
Это новый вид события, предназначенный специально для разработки модуля. При разработке модуля следует учитывать что скриптовой файл стал закрыт для других скриптовых файлов. Это хзначить что теперь можно использовать одноименные переменные и названия функций в разных скриптовых файлах и они не будут перезаписываться, но это накладывает ограничение: нельзя получить доступ к функции из другого скриптового файла простым способом. Для получения доступа к функции из другого скриптового файла, следует импортировать скриптовый файл с помощью команды module.Import("FileName.js").

5.2. Добавление в Store

Модуль, при добавлении из Store (магазина), запрашивает параметры устройства (IP, port и т.д). Например, HDL GPRS при добавлении из магазина запрашивает следующие параметры:



Для передачи этих параметров в модуль используется скрипт "Setup". Любой созданный скрипт может являться "Setup", для этого в параметрах скрипта нужно выставить флажок "true":



Например, скрипт "Setup" для модуля HDL выглядит следующим образом:

```
{
        // Data to connect drivers, created in Project Device Panel
        Drivers:
        [
      {
         Name: "HDL-BUS Pro Network (UDP)",
         Properties:
         [
```

```
            {
               Type: "textfield",
               Name: "Host",
               DefaultValue: "255.255.255.255",
               Validation: function(in_sValue) {
                  var l_aValueHost = in_sValue.split(".");
                  if (l_aValueHost.length != 4) {
                     return "Please input correct Host";
                  } else
                     if (parseInt(l_aValueHost[], 10)>=
                        && parseInt(l_aValueHost[], 10)<=255
                        && parseInt(l_aValueHost[1], 10)>=
                        && parseInt(l_aValueHost[1], 10)<=255
                        && parseInt(l_aValueHost[2], 10)>=
                        && parseInt(l_aValueHost[2], 10)<=255
                        && parseInt(l_aValueHost[3], 10)>=
                        && parseInt(l_aValueHost[3], 10)<=255
                     )
                        return
                     else
                        return "Please input correct Host";
               }
            },
            {
               Type: "textfield",
               Name: "Port",
               DefaultValue: "6000",
               Validation: function(in_sValue) {
                 if(parseInt(in_sValue, 10)>=)
                     return
                  else
                     return "Please input correct Port";
               }
            }
         ]
                }
        ],

        // General information for the module (driver and generated by
script)
        Module: [{
                Name: "Channels Count",
                Validation: function (in_sValue) {
                        if (parseInt(in_sValue, 10)>=)
                                return ;
                        else
                                return "Please input Count";
         }
      },
                {
         Type: "textfield",
```

```
                        Name : "SubnetID",
                        Validation : function (in_sValue) {
                                if (parseInt(in_sValue, 10)>=)
                                        return ;
                                else
                                        return "Please input SubnetID";
                        }
                },
      {
         Type: "textfield",
                        Name : "DeviceID",
                        Validation : function (in_sValue) {
                                if (parseInt(in_sValue, 10)>=)
                                        return ;
                                else
                                        return "Please input DeviceID";
                        }
                }

        ]
}
```

Файл Setup состоит из 2-ух частей. Первая часть это настройки драйверов. В этих настройках следует запрашивать ip адреса, порты и логины, для подключения драйвера у устройству. Вторая часть файла это общие настройки модуля, такие как количество выходов, ключи доступа. Для установки поля в Setup необходимо настроить следующие поля в скрипте:

- Тип поля - надо указать какое поле надо показать пользователю(текстовое поле, массив значений итд. Подробнее описано в API)
- Имя поля - название поля, которое увидит пользователь при настройке модуля
- Значение по умолчанию - какое значение следует подставлять по умолчанию
- Функция проверки - функция валидации значения в поле. Здесь требуется написать функцию, которая будет проверять корректность введенных данных.

И использовать их в скрипте модуля

```
module.GetProperty("Имя поля")
```

Помимо стандартных параметров, можно запрашивать любые, необходимые для инициализации модуля, параметры:

5.3. Создание SubDevice

Рассмотрим создание динамических под-устройств через скрипт на данном примере:

```
// Получаем уникальный идентификатор модуля и шины при запуске
 module.AddListener(IR.EVENT_MODULE_START, 0, function(){
   var netWorkName = "HDL-BUS Pro Network (UDP)"';
```

```
   // Берем существующее устройство и присваиваем переменной ''device''
   var device = module.getDevice(netWorkName);
   // Создаем новое под-устройство с параметрами:
   // Device - объект драйвера устройства iRidium
   // Имя создаваемого под-устройства, через которое будем к нему обращаться
и это же имя увидит пользователь
   var NewSubDevice = module.addSubDevice({
       Device: device,
       DeviceName: "HDL Dimmer 1"
    });
 });
```

При успешном создании под-устройства, в переменной NewSubDevice будет объект созданного под-устройства, у которого мы можем выполнить следующие операции:

- Работать с виджетами подустройства
- Работать с командами и тегами подустройства
- Работать с AEC(Actions, Events, Conditions)

5.4. Создание Commands и Feedback.

Существует два способа создания **Commands** (Команд) и **Feedback** (Фидбеков):

- Используя функции редактора iRidium Studio;
- С помощью скриптов.

Рассмотрим подробнее способ создания команд и фидбеков через код, но перед этим следует подключить модуль в дереве **PROJECT DEVICE PANEL**:

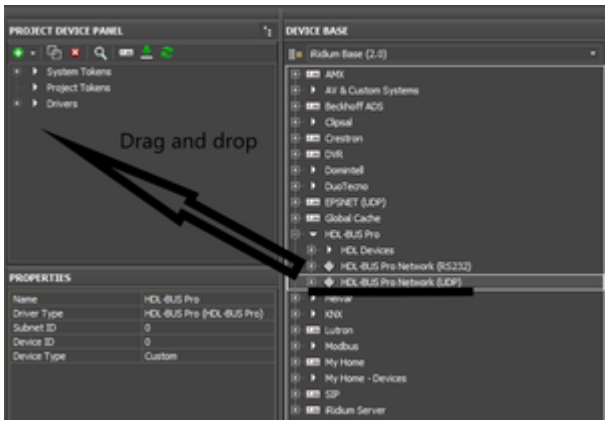


Рис. Подключение модуля

Для подключения достаточно перетащить и отпустить как показано на рисунке. Далее рассмотрим код:

```
l_oSubDevice = module.AddSubDevice("Light " + count, HDL_SERVER, false,
IR.SUB_DEVICE_TYPE_THROUGH_DIMMER);
```

```
// Adding channel StatusOnStart
if (count == 1)
 l_oSubDevice.AddChannel("Dimmer:statusOnStart",
[parseInt(SUBNET_ID.toString(16), 16), parseInt(DEVICE_ID.toString(16), 16),
HDL_CODES.singleChannelReadTarget, SEPARATOR, 0x0, 0x0, 0x0, 0x0, 0x1, 0x0,
0x1, 0xE8, 0x3, 0x0, 0x0]);

// Assigned to the variable name of the channel
var l_sChannelName = "Dimmer:channel" + count;

// Adding channels
l_oSubDevice.AddChannel(l_sChannelName, [parseInt(SUBNET_ID.toString(16),
16), parseInt(DEVICE_ID.toString(16), 16), HDL_CODES.singleChannelLigtning,
SEPARATOR, parseInt(count.toString(16), 16), 0x0, 0x0, 0x0, 0x01, 0x0, 0x0]);
l_oSubDevice.AddTag(l_sChannelName, [parseInt(SUBNET_ID.toString(16), 16),
parseInt(DEVICE_ID.toString(16), 16), HDL_CODES.singleChannelLigtning,
SEPARATOR, parseInt(count.toString(16), 16), 0x0, 0x0]);

// Adding smart virtual tag
l_oSubDevice.AddVirtualTag("Power", 0, false, IR.SUB_DEVICE_COMMAND_POWER,
true);
l_oSubDevice.AddVirtualTag("Power On", 0, false,
IR.SUB_DEVICE_COMMAND_POWER_ON, true);
l_oSubDevice.AddVirtualTag("Power Off", 0, false,
IR.SUB_DEVICE_COMMAND_POWER_OFF, true);
l_oSubDevice.AddVirtualTag("Level", 0, false, IR.SUB_DEVICE_COMMAND_LEVEL,
true);
```

Созданные команды и фидбеки через код **не показываются** в дереве PROJECT DEVICE PANEL.

5.5. Создание Widget и Remote

Для создания виджета через код и привязке его к определенному SubDevice нам понадобится создать шаблон этого виджета в редакторе. Виджет ограничен размерами:

- Ширина **640**
- Высота от **300**

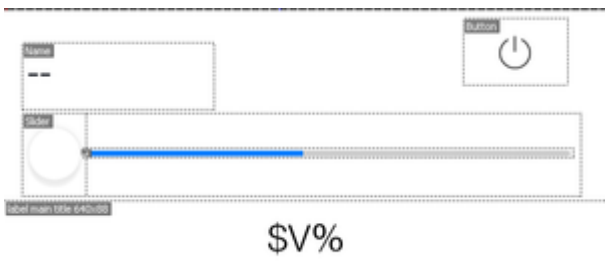


Рис. Пример виджета

Что бы указать программе, что данное всплывающее окно - не виджет, установите значение **False** в поле **Widget** в **Object Properties**.
`SubDevice.addWidget(`**`in_Widget`**`)`
Входным параметром метода является: **in_Widget** - объект - всплывающее окно, заранее созданный в редакторе.
Выходным параметром метода является: успешно или нет (**True**, **False**).
К примеру у нас 3 лампочки и для того, чтобы не создавать для каждой лампочки отдельный виджет, мы воспользуемся уже готовым примером нашего всплывающего окна и просто будем клонировать. Напишем пример создания виджета с помощью кода по уже существующему образцу.
Метод Clone позволяет клонировать уже существующее всплывающее окно.
`Module.ClonePopup (in_Popup, in_Name)`
Входные параметры метода:

- **in_Name** - имя нового виджета.
- **in_Popup** - ссылка на всплывающее окно.

Выходным параметром метода является: успешно или нет (**True**, **False**).

```
// Обращаемся к виджету который собираемся скопировать
 module = B.getModule(moduleID)
 var popup = module.GetPopup("Dimmer");
 for (var i = 1; i<=3; i++)
 {  // Создаем виджет методом клонирования
 var widget = NewSubDevice.addWidget(Module.ClonePopup(popup, "Dimmer"+ i));
 }
```

Для диммера добавим функцию, которая позволяет двигать слайдер:

```
// пользовательский слайдер для уровня
 function UserSlider(Level, Slider)
 {
    Property = "X";
    Len = "Width";
    // Функция вычисления позиции слайдера относительно уровня
    function Move(){
       Slider[Property] = Level.Value * (Level[Len] -50) / 100;
    }
    // Подписка на события
    module.AddListener(IR.EVENT_ITEM_PRESS, Level, Move); // нажатие на
уровень
    module.AddListener(IR.EVENT_MOUSE_MOVE, Level, Move); // движение мыши по
уровню
    module.AddListener(IR.EVENT_TOUCH_MOVE, Level, Move); // движение пальца
по уровню
    module.SetInterval(600, Move);  // автообновление через 600 мс
 };
```

Имена новых виджетов не должны повторяться.

5.6. Создание связи между графическим элементом виджета и каналом драйвера.

Связь между графическим элементом и каналом драйвера образуется при помощи связи канала обратной связи драйвера (Feedback) и свойства графического элемента на созданном виджете. Напишем пример создания связей между графическими элементами созданного виджета и каналов драйверов с помощью кода:
`module.AddRelation(in_Feedback, in_Graphic);`
Входным параметром метода является:
in_Feedback - полный путь к каналу обратной связи в строковом типе (пример: "Drivers.HDL-BUS Pro Network (UDP)" + ".HDL-MC48IPDMX.231:channel").
in_Graphic - полный путь к свойству графического элемента в строковом типе (пример: "UI.Dimmer" + ".level.Value").
Выходным параметром метода является: успешно или нет (**True**, **False**).

5.7. Адаптация модуля для мобильных устройств.

Так как каждый модуль является кроссплатформенным, то при разработке модуля надо подготовить графическую часть в 2-ух вариациях - Планшетной и телефонной. Подготовка означает что надо создать Remote как для планшета, так и для смартфона. После создания экземпляров Remote, необходимо написать скрипт, который будет определять текущую версию панели(планшет или телефон) и показывать указанный попап.

```
 if (IR.DisplayType == IR.DISPLAY_TYPE_PHONE) { //Если модуль открыт на
телефоне
    var l_oPopupScanner = module.GetPopup("Phone"); //Показываем попап,
отрисованный для телефона
  }
 else {
   var l_oPopupScanner = module.GetPopup("Tablet");//Если планшет, то
открываем попап для планшета
 }
```

5.9. Создание сканера

Scanner (сканер) - это модуль, который анализирует какую-либо шину на подключённые устройства. Подключенные устройства формируются в список, для выбранного устройства качается модуль и становится доступен в i3 lite.
Структура работы сканера:

1. Сканер загружается из магазина;
2. Пользователь вводит параметры сканера (IP, Port и т.д) и эти параметры передаются в

скрипт сканера;
3. Сканер, обработав введенные параметры, опрашивает шину на устройства;
4. Из полученных устройств формируется список;
5. При выборе определенного устройства в списке, сканер передаёт параметры выбранного устройства в модуль.

При разработке сканера нужно учесть:

- Параметры сканера, при добавлении из магазина;
- Логику сканера;
- Передачу параметров выбранного устройства в модуль;
- Логику модуля, с учётом переданных от сканера параметров.

Процесс разработки сканера ничем не отличается от разработки модуля. Точно также сканеру необходимо нарисовать визуальную часть, создать драйвер и написать скрипт с логикой. Основным отличием является то, что сканер может работать только на панели управления и сканер никогда не будет запущен на сервере, поэтому, при разработке скриптов, нет необходимости разделять скрипты на драйверные и интерфейсные.
Также при разработке сканера появляется новая логика работы. Первым делом необходимо сделать модуль для управления устройством, далее надо загрузить модуль в магазин модулей. При загрузке модуля, вы увидите уникальный идентификатор вашего модуля. Когда вы будете разрабатывать сканер, вам необходимо написать скрипт со следующей логикой:

1. Создание драйвера
2. Поиск оборудования
3. Если оборудование найдено, надо определить что это за оборудование
4. При помощи команды ModuleSetupFinish (ID модуля, js объект с настройками setup данного модуля)

После этого приложение скачет модуль из магазина модулей и установит его.

Загрузка модулей в магазин модулей

Модуль можно загрузить в iRidium Store для этого нужно:

1. Перейти в личный кабинет разработчика модулей по адресу http://www.iridiummobile.net/approve/dev/
2. Нажать кнопку “Добавить новый модуль”, ввести имя модуля и указать его тип(сканер или модуль)
3. Ввести информацию о модуле
4. Загрузить файл модуля в Development, если вы загружаете тестовую версию модуля
5. Загрузить файл модуля в Production, если вы хотите открыть модуль для всех и начать продавать его
6. Дождаться подтверждения работоспособности модуля от модератора
7. Модуль появится в магазине модулей